Sorting ..

**1. Quick Sort**

#include <iostream>

using namespace std;

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = low - 1;

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr[i], arr[j]);

}

}

swap(arr[i + 1], arr[high]);

return i + 1;

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

int main() {

int arr[] = {10, 7, 8, 9, 1, 5};

int n = sizeof(arr) / sizeof(arr[0]);

quickSort(arr, 0, n - 1);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

**2. Merge Sort**

#include <iostream>

using namespace std;

void merge(int arr[], int l, int m, int r) {

int n1 = m - l + 1;

int n2 = r - m;

int \*L = new int[n1];

int \*R = new int[n2];

for (int i = 0; i < n1; i++) L[i] = arr[l + i];

for (int j = 0; j < n2; j++) R[j] = arr[m + 1 + j];

int i = 0, j = 0, k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

delete[] L;

delete[] R;

}

void mergeSort(int arr[], int l, int r) {

if (l < r) {

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int n = sizeof(arr) / sizeof(arr[0]);

mergeSort(arr, 0, n - 1);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

**3. Bubble Sort**

#include <iostream>

using namespace std;

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

int arr[] = {64, 34, 25, 12, 22, 11, 90};

int n = sizeof(arr) / sizeof(arr[0]);

bubbleSort(arr, n);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

**4. Radix Sort**

#include <iostream>

using namespace std;

int getMax(int arr[], int n) {

int max = arr[0];

for (int i = 1; i < n; i++) {

if (arr[i] > max) max = arr[i];

}

return max;

}

void countingSort(int arr[], int n, int exp) {

int \*output = new int[n];

int count[10] = {0};

for (int i = 0; i < n; i++) count[(arr[i] / exp) % 10]++;

for (int i = 1; i < 10; i++) count[i] += count[i - 1];

for (int i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

for (int i = 0; i < n; i++) arr[i] = output[i];

delete[] output;

}

void radixSort(int arr[], int n) {

int max = getMax(arr, n);

for (int exp = 1; max / exp > 0; exp \*= 10) {

countingSort(arr, n, exp);

}

}

int main() {

int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};

int n = sizeof(arr) / sizeof(arr[0]);

radixSort(arr, n);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}